Space Details

Key:	OneStopShopping
Name:	One-Stop Shopping for MBARI Upper Water Column Data
Description:	2009 Project to improve infrastructure for accessing MBARI upper water column data.
Creator (Creation Date):	mccann (Jan 13, 2009)
Last Modifier (Mod. Date):	mccann (Jan 14, 2009)

Available Pages

- Home
  - Analyzing signals from MARS using SSDS and Matlab
  - dods server transition
  - Examples using pydap from Python to access BOG data via DRDS
  - Google Earth dynamic queries
  - installing DRDS and connecting to BOG
  - Project kick-off meeting
    - John Ryan's 2 Use Cases
  - Project meeting on 10 September 2009
  - Project meeting on 12 November 2009
  - Project meeting on 19 February 2009
  - Project meeting on 2 April 2009
  - Project meeting on 7 May 2009
  - Systems Engineering Information
  - Upper Water Column Data
  - Upper Water Column Data access web page
- .bookmarks

Home

This page last changed on Feb 03, 2010 by mccann.

2009 Project: One-stop Shopping for MBARI Upper Water Column Data

Notes from proposal process in 2008

- 25 June 2008 "kick-off" meeting (notes)
- Abstract submitted (note the intention to submit as part of the MDUC Core data project proposal)
- Presentation given to Management Team on 8 July 2008
- Management Team feedback on 11 July 2008
- One-stop_shopping_2009_WBS.xls(original, deprecated)
- Modified WBS (moved Brian's time to R2) spreadsheet - 6 Feb 2009
- Project Proposal (draft) - 15 July 2008
- Approved Project Proposal (on PCO web site)

Meeting notes 2009

- Project kick-off meeting on 14 Jan 2009
- Project meeting on 19 February 2009 10:00 AM
- Project meeting on 2 April 2009 10:00 AM
- Project meeting on 7 May 2009 2:00 PM
- Project meeting on 10 September 2009 10:00 AM

Miscellaneous Documents

- Systems Engineering Information and implementation plan
- Overall Upper Water Column Data access web page (to become the public access to MBARI Upper Water Column data)
- Analyzing signals from MARS using SSDS and Matlab
- Upper Water Column Data Descriptions
- Setup for Google Earth dynamic queries
- Setup for installing DRDS and connecting to BOG
- BOG Standard Products: Time Series
- BOG Standard Products: Profiles Casts (Core data)
- BOG Standard Products: SECRET Line 67 Depth Contours
- BOG Database: Key Fields, SQL examples, Table descriptions
- Design Principles for Science Web Sites (Powerpoint presentation from ESSI Workshop, Aug 3-5, 2009)
- Data Access webpage

Client Access Examples

Note that some examples are also shown in the meeting notes from April and May

- Examples using pydap from Python to access BOG data via DRDS
- Examples using ncdataset from Matlab to access OPeNDAP data

Deliverables

- Data Access webpage
- TDS Aggregation server - MBARI Mooring data
- Matlab toolkit - ncdataset - for accessing TDS data and writing NetCDF files

Analyzing signals from MARS using SSDS and Matlab

This page last changed on Jan 13, 2009 by mccann.

Following is an example of using Matlab and the loaddap tool to analyze high frequency data from the MARS2008 CTD instrument.

The code below will produce power spectra density plots from which signals may be discovered. We may also use this exercise to help us adjust sampling frequencies so proper experiment design and efficient use of MBARI's limited infrastructure.

```
% Load Original netCDF file into Matlab workspace.  URL may be discovered from the netcdf links on
the web page:
% http://dods.mbari.org/data/ssdsdata/deployments/mars2008/current_qcPlots.html

loaddap('http://dods.mbari.org/cgi-bin/nph-nc/data/ssdsdata/deployments/mars2008/200706/
mars2008_ctd1613_20081110_original.nc')

% May now do a 'who' to see what structures have been loaded.  Extract temp and time vectors from
the SSDS data set:
temp = squeeze(Temperature.Temperature);
time = squeeze(Temperature.esecs);

% Construct regular 10-second interval time series for generating the spectral analysis
dT = 10;
time10 = [time(1):dT:time(end)];
temp10 = interp1(time,temp,time10);

% Apply 10% Cosine window to the time series and compute fft from demeaned temp time series
nt = length(time10);
rampUp = sin(linspace(0,pi/2,0.05*nt));
rampDown = cos(linspace(0,pi/2,0.05*nt));
nMiddle = nt - length(rampUp) - length(rampDown);
cos10 = [rampUp ones(1,nMiddle) rampDown];
temp_fft = fft(cos10 .* (temp10/mean(temp10)-1));

% Construct frequency axis and plot
f = linspace(0,1/dT,nt);
loglog(f, abs(temp_fft));
title('Time Series analysis of MARS CTD Instrument')
xlabel('Frequency (Hz)')
ylabel('PSD (Temperature)')
print -dpng temp_pds.png
```

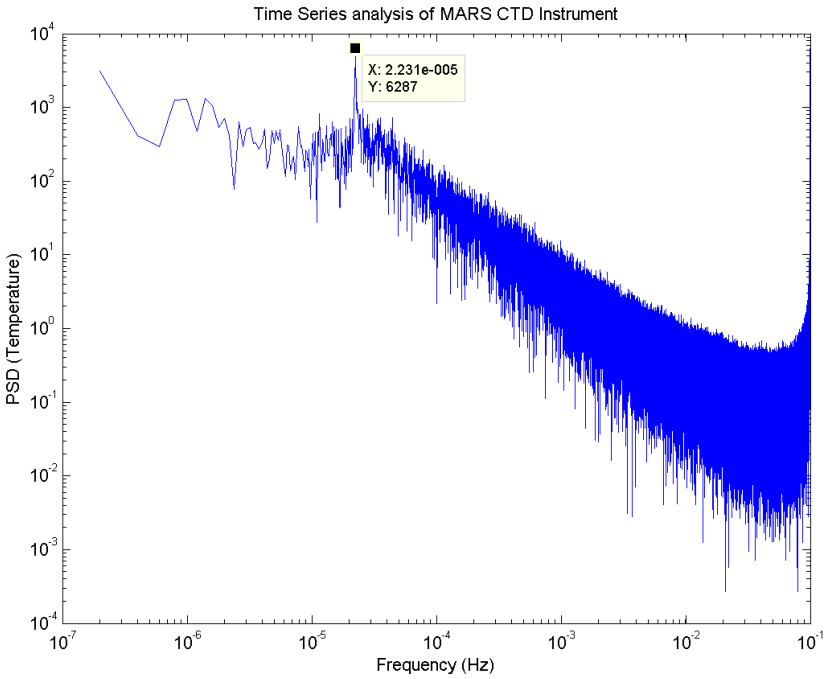


Figure1. Power spectral density for Temperature at the MARS observatory. The peak in the power spectral density at 2.23 E-5 Hz is the M2 tide at 12.4 hours. Power density increases at high frequencies (periods less than 20 seconds) perhaps due to instrument noise.

Let's do the same for pressure:

```
pres = squeeze(Pressure.Pressure);
pres10 = interp1(time,pres,time10);
pres_fft = fft(cos10 .* (pres10/mean(pres10)-1));
loglog(f, abs(pres_fft));
title('Time Series analysis of MARS CTD Instrument')
xlabel('Frequency (Hz)')
ylabel('PSD (Pressure)')

% Use "peak<n> = ginput" in Matlab session to identify the peaks and label them with:
text(peak1(1), peak1(2) , sprintf('%.1f hours', 1/peak1(1)/3600), 'fontsize', 5)
text(peak2(1), peak2(2) , sprintf('%.1f hours', 1/peak2(1)/3600), 'fontsize', 5)
text(peak3(1), peak3(2) , sprintf('%.1f hours', 1/peak3(1)/3600), 'fontsize', 5)
print -dpng pres_pds.png
```

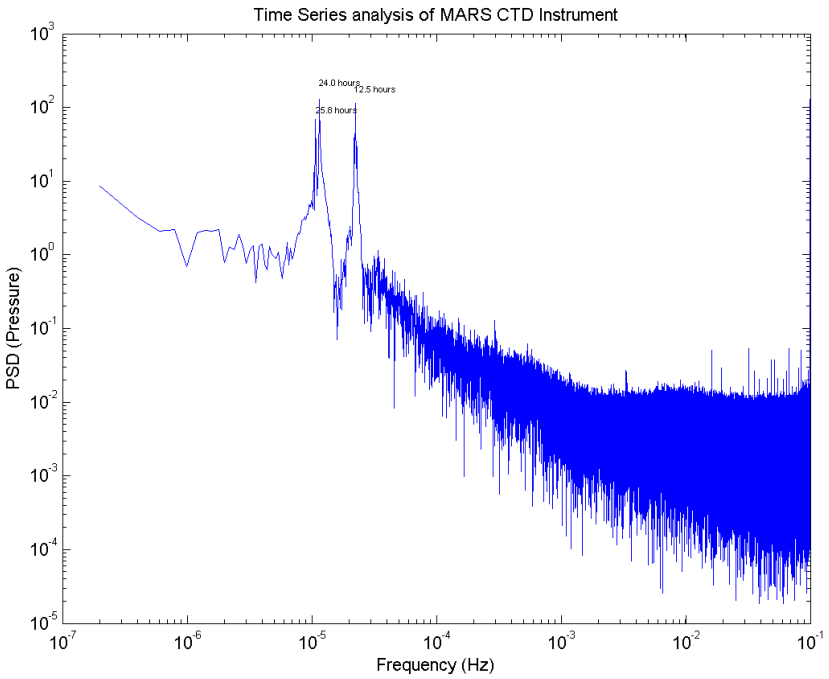


Figure 2. Power spectral density for Pressure at the MARS observatory. The sharp peaks at 25.8, 24.0, and 12.5 are due to the O1 (Principal diurnal lunar), P1 (Principal diurnal solar), and M2 (Principal semidiurnal lunar) tides. There is a shoulder of higher power density in the 10 to 1000 second band that we do not see in the temperature spectra.

dods server transition

This page last changed on May 11, 2009 by mccann.

The server archaea (aka dods and searc) serves a lot of data. Here is a list of URLs that need to continue to work after elvis takes over its function:

1. http://dods.mbari.org/cgi-bin/nph-nc/data/ssdsdata/ssds/generated/netcdf/files/ssds.shore.mbari.org/auvctd/missionlogs/2009/2009122/2009.122.02/navigation.nc
2.

This page last changed on Jan 08, 2013 by mccann.

Examples using pydap from Python

NOTE: The DRDS web app must be installed in Tomcat on elvis for this to work. This was last done under apache-tomcat-6.0.18 and the current Tomcat on elvis is apache-tomcat-6.0.29. The example illustrated below is for getting data from just one table from BOG. There is a considerable overhead with configuring DRDS (https://oceana.mbari.org/confluence/display/OneStopShopping/installing+DRDS+and+connecting+to+BOG) and there are more direct ways to access all BOG data from Python using SQL or STOQS. - Mike McCann 8 Jan 2013.

Please see http://pydap.org/client.html#accessing-sequential-data for where these examples came from. Below are examples accessing our data from elvis.

- Print out bottles where DEPTH is greater than 4300
- Print out bottles where SAL is less than 28
- Plot profiles of bottle data from near mooring M2 (a slightly more complex example using the Matplotlib package)

Print out bottles where DEPTH is greater than 4300:

Copy-n-paste into Python session:

```
from pydap.client import open_url                                    # Use PyDAP client library
from http://pydap.org/
dataset = open_url('http://elvis.shore.mbari.org/dods/bog/BCTD')     # Read data directly from
the database via OPeNDAP/DRDS
print dataset.BCTD.keys()                                            # Print column names
my_bottles = dataset.BCTD[(dataset.BCTD.DEPTH > 4300)]               # Select bottles that are
deeper than 4000
print len(my_bottles['DEPTH'])                                       # Print length of
my_bottles list
for d,la,lo,id,t,o in zip(my_bottles.DEPTH, my_bottles.DEC_LAT, my_bottles.DEC_LONG,
my_bottles.SITE_ID, my_bottles.TMP, my_bottles.O2):
print d,la,lo,id,t,o
```

What it looks like when you execute the above:

```
<106 elvis.shore.mbari.org /u/mccann> python
Python 2.4.3 (#1, Jun 11 2009, 14:09:58)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-44)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from pydap.client import open_url                                 # Use PyDAP client
library from http://pydap.org/
>>> dataset = open_url('http://elvis.shore.mbari.org/dods/bog/BCTD')  # Read data directly
from the database via OPeNDAP/DRDS
>>> print dataset.BCTD.keys()                                         # Print column names
['CTD_BOTTLE_ID', 'CRUISE', 'EXPD', 'SEQ', 'DEPTH', 'BOTTLE', 'PROJECT', 'CTRB_ID', 'PLATFORM',
'RJDAY', 'DATE_TIME', 'JTIME', 'YEAR_DATE',
'DEC_LAT', 'DEC_LONG', 'SITE_ID', 'PRESSURE', 'CHL_GFF', 'PHAEO_GFF', 'FOFA_GFF', 'VF_GFF',
'VE_GFF', 'SENS_GFF', 'FO_GFF', 'FA_GFF', 'FLUOR_RTO',
'CHL_1u', 'PHAEO_1u', 'FOFA_1u', 'VF_1u', 'VE_1u', 'FO_1u', 'FA_1u', 'SENS_1u', 'CHL_5u',
'PHAEO_5u', 'FOFA_5u', 'VF_5u', 'VE_5u', 'SENS_5u', 'FO_5u',
'FA_5u', 'TMP', 'SAL', 'CONDUCT', 'SIG_T', 'PARCOS', 'PAR4PI', 'TRANSMISS', 'CHLA', 'FLUOR', 'NH4',
'NO2', 'NO3', 'SIO4', 'PO4', 'TCO2', 'O2',
'OXY_ML', 'OXY_PS', 'IRRD_490', 'IRRD_520', 'IRRD_555', 'ALTIMETER', 'rdep', 'sal2', 'temp2',
'cond2', 'pot_tmp', 'OxyT', 'OxyC', 'TransV',
'TransBeam', 'POT_TMP2', 'OxyV', 'ISUS_NO3']
>>> my_bottles = dataset.BCTD[(dataset.BCTD.DEPTH > 4300)]             # Select bottles that
are deeper than 4000
```

```
>>> print len(my_bottles['DEPTH'])                                          # Print length of
my_bottles list
18
>>> for d,la,lo,id,t,o in zip(my_bottles.DEPTH, my_bottles.DEC_LAT, my_bottles.DEC_LONG,
my_bottles.SITE_ID, my_bottles.TMP, my_bottles.O2):
...    print d,la,lo,id,t,o
...

4500 35.441 -124.89 67-90 1.5242 nan
4500 35.457 -124.897 67-90 1.5183 nan
4400 35.453 -124.901 67-90 1.5193 nan
4500 35.453 -124.903 67-90 1.5199 nan
4400 35.454 -124.905 67-90 1.5138 nan
4370 35.459 -124.907 67-90 1.5212 nan
4465 36.576 -125.741 60-90 1.5342 nan
4380 35.467 -124.94 67-90 1.5182 nan
4500 35.439 -124.89 67-90 1.5211 nan
4335 35.458 -124.907 67-90 1.5175 nan
4400 35.446 -124.919 67-90 1.5187 nan
4400 35.45 -124.902 67-90 1.5146 nan
4450 35.122 -125.605 67-100 1.5294 nan
4450 35.122 -125.605 67-100 1.5296 nan
4400 35.462 -124.91 67-90 1.5247 nan
4400 35.455 -124.909 67-90 1.516 nan
4500 35.453 -124.903 67-90 1.5253 nan
4400 35.452 -124.911 67-90 1.515 nan
>>>
```

Print out all bottle data with Salinity less than 28:

Copy-n-paste into Python session:

```
from pydap.client import open_url                                          # Use PyDAP client library
from http://pydap.org/
dataset = open_url('http://elvis.shore.mbari.org/dods/bog/BCTD')           # Read data directly from
the database via OPeNDAP/DRDS
print dataset.BCTD.keys()                                                  # Print column names
mb = dataset.BCTD[(dataset.BCTD.SAL < 28)]                                 # Select bottles where
Salinity is less than 28
print len(mb['DEPTH'])                                                     # Print length of
my_bottles list
outText = "DATE_TIME, DEPTH, DEC_LAT, DEC_LONG, SITE_ID, TMP, SAL\n"       # Column headers
for dt,d,la,lo,id,t,s in zip(mb.DATE_TIME, mb.DEPTH, mb.DEC_LAT, mb.DEC_LONG, mb.SITE_ID, mb.TMP,
mb.SAL):
outText = outText + "%s, %s, %s, %s, %s, %s, %s\n" % (dt, d, la, lo, id, t, s)          # Collect the
data

print outText
```

What it looks like when you execute the above:

```
>>> from pydap.client import open_url                                      # Use PyDAP client
library from http://pydap.org/
>>> dataset = open_url('http://elvis.shore.mbari.org/dods/bog/BCTD')       # Read data directly
from the database via OPeNDAP/DRDS
>>> print dataset.BCTD.keys()                                              # Print column names
['CTD_BOTTLE_ID', 'CRUISE', 'EXPD', 'SEQ', 'DEPTH', 'BOTTLE', 'PROJECT', 'CTRB_ID', 'PLATFORM',
'RJDAY', 'DATE_TIME', 'JTIME',
'YEAR_DATE', 'DEC_LAT', 'DEC_LONG', 'SITE_ID', 'PRESSURE', 'CHL_GFF', 'PHAEO_GFF', 'FOFA_GFF',
'VF_GFF', 'VE_GFF', 'SENS_GFF',
'FO_GFF', 'FA_GFF', 'FLUOR_RTO', 'CHL_1u', 'PHAEO_1u', 'FOFA_1u', 'VF_1u', 'VE_1u', 'FO_1u',
'FA_1u', 'SENS_1u', 'CHL_5u',
'PHAEO_5u', 'FOFA_5u', 'VF_5u', 'VE_5u', 'SENS_5u', 'FO_5u', 'FA_5u', 'TMP', 'SAL', 'CONDUCT',
'SIG_T', 'PARCOS', 'PAR4PI',
```

```
'TRANSMISS', 'CHLA', 'FLUOR', 'NH4', 'NO2', 'NO3', 'SIO4', 'PO4', 'TCO2', 'O2', 'OXY_ML', 'OXY_PS',
'IRRD_490', 'IRRD_520',
'IRRD_555', 'ALTIMETER', 'rdep', 'sal2', 'temp2', 'cond2', 'pot_tmp', 'OxyT', 'OxyC', 'TransV',
'TransBeam', 'POT_TMP2',
'OxyV', 'ISUS_NO3']
>>> mb = dataset.BCTD[(dataset.BCTD.SAL < 28)]                              # Select bottles where
Salinity is less than 28
>>> print len(mb['DEPTH'])                                                  # Print length of
my_bottles list
19
>>> outText = "DATE_TIME, DEPTH, DEC_LAT, DEC_LONG, SITE_ID, TMP, SAL\n"     # Column headers
>>> for dt,d,la,lo,id,t,s in zip(mb.DATE_TIME, mb.DEPTH, mb.DEC_LAT, mb.DEC_LONG, mb.SITE_ID,
mb.TMP, mb.SAL):
...    outText = outText + "%s, %s, %s, %s, %s, %s, %s\n" % (dt, d, la, lo, id, t, s)        #
Collect the data
...
>>> print outText
DATE_TIME, DEPTH, DEC_LAT, DEC_LONG, SITE_ID, TMP, SAL
2007-01-28 02:15:37.0, 10, 37.555, -122.191, 38, 9.0597, 27.5771
2007-01-28 02:15:37.0, 10, 37.555, -122.191, 38, 9.0607, 27.5745
2007-01-28 02:42:04.0, 10, 37.569, -122.219, 39, 9.093, 27.7954
2007-01-28 02:42:04.0, 10, 37.569, -122.219, 39, 9.0929, 27.7957
2007-01-28 03:04:38.0, 10, 37.58, -122.244, 40, 9.1872, 27.994
2007-01-28 03:04:38.0, 10, 37.58, -122.244, 40, 9.1877, 27.9945
2007-01-28 11:13:18.0, 0, 37.978, -122.43, 54, 9.1868, 25.3831
2007-01-28 11:36:34.0, 0, 38.007, -122.404, 55, 9.2646, 25.4848
2007-01-28 11:57:32.0, 5, 38.028, -122.37, 56, 9.3301, 26.1335
2007-01-28 11:57:32.0, 5, 38.028, -122.37, 56, 9.3298, 26.1474
2007-01-28 12:53:28.0, 20, 38.062, -122.263, 58, 9.1294, 24.1165
2007-01-28 12:53:28.0, 20, 38.062, -122.263, 58, 9.1296, 24.1172
2007-01-28 12:28:13.0, 5, 38.051, -122.313, 57, 9.2559, 25.5208
2007-01-28 12:28:13.0, 5, 38.051, -122.313, 57, 9.2593, 25.5637
2006-04-17 08:33:43.0, 10, 47.775, -124.648, 13.3 -2.0, 10.22, 27.4995
2006-04-17 08:33:43.0, 5, 47.775, -124.648, 13.3 -2.0, 10.2026, 27.4343
2006-04-17 08:33:43.0, 0, 47.775, -124.648, 13.3 -2.0, 10.1847, 27.4071
2006-04-19 00:13:59.0, 5, 46.429, -124.222, 20.0 3.0, 10.3676, 23.8885
2006-04-19 00:13:59.0, 0, 46.429, -124.222, 20.0 3.0, 10.8401, 21.1543
```

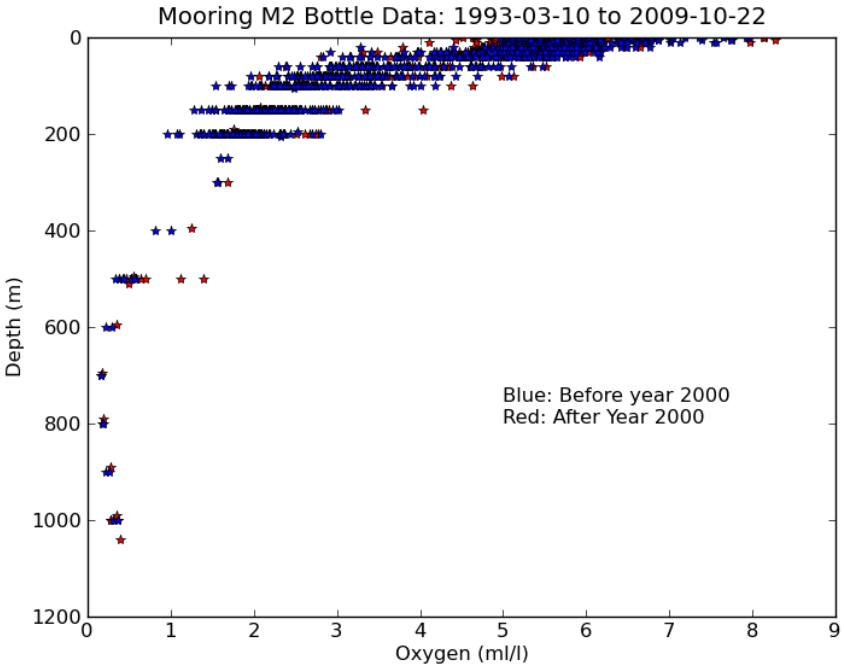
Plot profiles of bottle data from near mooring M2 (a slightly more complex example using the Matplotlib package):

Copy-n-paste into Python session:

```
from pydap.client import open_url                                        # Use PyDAP client library
from http://pydap.org/
from pylab import *                                                      # Use Matlab-like plotting
module
dataset = open_url('http://elvis.shore.mbari.org/dods/bog/BCTD')         # Read data directly from
the database via OPeNDAP/DRDS
# From using Google Earth to get bounding box around M2's location:
BBOX=-122.413310,36.673487,-122.357704,36.7042573
m2bottles = dataset.BCTD[
(dataset.BCTD.DEC_LONG > -122.413310) &
(dataset.BCTD.DEC_LONG < -122.357704) &
(dataset.BCTD.DEC_LAT > 36.673487) &
(dataset.BCTD.DEC_LAT < 36.7042573) ]

# Loop through query results and build dictionary keyed on profile (DATE_TIME is a useful proxy)
# Create a dictionaries of lists for thess data structures
depthList = dict()
oxyList = dict()
for dt in m2bottles.DATE_TIME:
depthList[dt] = list()
oxyList[dt] = list()
```

```
# Now append to these lists.  This creates the 'arrays' that we can pass to the plot routine.
for (dt,d,o) in (zip(m2bottles.DATE_TIME, m2bottles.DEPTH, m2bottles.OXY_ML)):
depthList[dt].append(d)
oxyList[dt].append(o)

# Order the time list
dtList = depthList.keys()
dtList.sort()

# Plot point from the profiles in time order
# To make it a little interesting color code the points for before and after the year 2000
fig = figure()
for dt in dtList:
print "Plotting profile collected on " + dt
print oxyList[dt], depthList[dt]
if int(dt[0:4]) > 2000:
plot(oxyList[dt], depthList[dt], 'b*')
else:
plot(oxyList[dt], depthList[dt], 'r*')
#axis([0, 9, 0, 1200])
ax = gca()
ax.set_ylim(ax.get_ylim()[::-1])        # Reverse the depth axis


xlabel('Oxygen (ml/l)')
ylabel('Depth (m)')
title("Mooring M2 Bottle Data: " + dtList[0].split(' ')[0] + " to " + dtList[-1].split(' ')[0])
text(5, 800, 'Blue: Before year 2000\nRed: After Year 2000')

show()

fig.savefig("M2_oxy.png")
```

Produces this figure:



And the output of the above set of python commands:

```
<121 elvis.shore.mbari.org /u/ssdsadmin> python
```

```
Python 2.4.3 (#1, Jun 11 2009, 14:09:58)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-44)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from pydap.client import open_url                                    # Use PyDAP client
library from http://pydap.org/
>>> from pylab import *                                                  # Use Matlab-like
plotting module
>>> dataset = open_url('http://elvis.shore.mbari.org/dods/bog/BCTD')     # Read data directly
from the database via OPeNDAP/DRDS
>>> # From using Google Earth to get bounding box around M2's location:
BBOX=-122.413310,36.673487,-122.357704,36.7042573
... m2bottles = dataset.BCTD[
...    (dataset.BCTD.DEC_LONG > -122.413310) &
...    (dataset.BCTD.DEC_LONG < -122.357704) &
...    (dataset.BCTD.DEC_LAT > 36.673487) &
...    (dataset.BCTD.DEC_LAT < 36.7042573) ]
>>>
>>> # Loop through query results and build dictionary keyed on profile (DATE_TIME is a useful
proxy)
... # Create a dictionaries of lists for thess data structures
... depthList = dict()
>>> oxyList = dict()
>>> for dt in m2bottles.DATE_TIME:
...    depthList[dt] = list()
...    oxyList[dt] = list()
...
>>> # Now append to these lists.  This creates the 'arrays' that we can pass to the plot routine.
... for (dt,d,o) in (zip(m2bottles.DATE_TIME, m2bottles.DEPTH, m2bottles.OXY_ML)):
...    depthList[dt].append(d)
...    oxyList[dt].append(o)
...
>>> # Plot point from the profiles in time order, color code according to date
... dtList = depthList.keys()
>>> dtList.sort()
>>>
>>> fig = figure()
>>> for dt in dtList:
...    print "Plotting profile collected on " + dt
...    print oxyList[dt], depthList[dt]
...    if int(dt[0:4]) > 2000:
...       plot(oxyList[dt], depthList[dt], 'b*')
...    else:
...       plot(oxyList[dt], depthList[dt], 'r*')
...    #axis([0, 9, 0, 1200])
...    ax = gca()
...    ax.set_ylim(ax.get_ylim()[::-1])         # Reverse the depth axis
...
Plotting profile collected on 1993-03-10 21:46:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9ac3f8c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1993-07-21 19:41:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9ada84c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1993-10-06 19:50:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9adaa4c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1993-10-27 18:46:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9adac4c>]
```

```
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1993-11-23 21:18:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9adae4c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1993-12-15 21:01:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 10, 20, 30, 40, 60, 80, 100, 150, 200]
[<matplotlib.lines.Line2D object at 0x9adaeac>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1993-12-29 18:23:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9ae326c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1994-01-26 20:03:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 200]
[<matplotlib.lines.Line2D object at 0x9ae346c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1994-02-23 19:03:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9ae366c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1994-03-09 19:53:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9ae386c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1994-03-23 20:45:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9ae3a6c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1994-04-13 18:44:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9ae3c6c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1994-05-18 18:12:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [5, 10, 20, 30, 40, 60, 80, 100, 150, 200,
0]
[<matplotlib.lines.Line2D object at 0x9ae3e6c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1994-06-29 18:50:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 80, 100, 150, 200]
[<matplotlib.lines.Line2D object at 0x9ae3ecc>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1994-07-20 18:57:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9aeb28c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1994-08-11 20:01:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9aeb48c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1994-09-28 19:43:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9aeb68c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1994-10-19 18:33:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
```

```
[<matplotlib.lines.Line2D object at 0x9aeb88c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1994-12-21 19:02:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9aeba8c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1995-02-09 00:03:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9aebc8c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1995-03-01 20:27:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 40, 60, 80, 100, 150, 200]
[<matplotlib.lines.Line2D object at 0x9aebe8c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1995-03-15 19:09:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9aebeec>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1995-03-29 20:43:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [10, 20, 30, 40, 60, 80, 100, 150, 200, 0,
5]
[<matplotlib.lines.Line2D object at 0x9af22ac>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1995-04-23 04:12:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9af24ac>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1995-05-02 20:27:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9af26ac>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1995-05-24 19:52:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9af28ac>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1995-06-14 21:06:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9af2aac>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1995-06-28 18:09:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9af2cac>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1995-07-12 18:45:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [20, 30, 40, 60, 80, 100, 150, 200, 0, 5,
10]
[<matplotlib.lines.Line2D object at 0x9af2eac>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1995-07-26 20:24:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 200]
[<matplotlib.lines.Line2D object at 0x9af2f0c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1995-08-09 18:46:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 200]
[<matplotlib.lines.Line2D object at 0x9b422cc>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1995-08-23 17:57:00.0
```

```
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b424cc>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1995-09-13 20:36:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b426cc>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1995-09-27 18:58:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b428cc>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1995-10-11 19:18:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b42acc>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1995-10-25 19:01:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b42ccc>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1995-11-15 19:40:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b42ecc>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1996-01-03 19:46:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b42f2c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1996-02-14 20:53:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b4a2ec>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1996-04-03 20:41:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b4a4ec>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1996-06-06 00:35:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 80, 100, 150, 200]
[<matplotlib.lines.Line2D object at 0x9b4a6ec>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1996-08-06 18:57:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b4a8ec>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1996-08-27 19:51:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b4aaec>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1996-09-17 19:29:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b4acec>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1996-10-09 18:37:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
```

```
[<matplotlib.lines.Line2D object at 0x9b4aeec>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1996-10-30 20:40:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b4af4c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1997-03-04 22:30:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 40, 60, 80, 100, 150, 700,
200]
[<matplotlib.lines.Line2D object at 0x9b5230c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1997-05-07 19:02:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 15, 20, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b5250c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1997-06-03 13:45:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [100, 150, 200, 300, 400, 500, 600,
700, 800, 900, 1000, 80]
[<matplotlib.lines.Line2D object at 0x9b5270c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1997-06-03 15:09:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 0, 5, 5, 10, 10, 20, 20, 30, 30,
40, 60]
[<matplotlib.lines.Line2D object at 0x9b5290c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1997-06-18 19:31:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 60, 80, 100, 200, 500]
[<matplotlib.lines.Line2D object at 0x9b52b0c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1997-07-29 11:25:00.0
[nan, nan, nan, nan] [0, 0, 1000, 1000]
[<matplotlib.lines.Line2D object at 0x9b52d0c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1997-07-30 18:33:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b52f0c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1997-08-11 19:28:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [5, 10, 20, 30, 40, 60, 80, 100, 150, 200,
0]
[<matplotlib.lines.Line2D object at 0x9b52f6c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1997-10-22 18:33:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b5932c>]
(0.059999999999999998, -0.059999999999999998)
Plotting profile collected on 1997-11-12 19:28:00.0
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan] [0, 5, 10, 20, 30, 40, 60, 80, 100, 150,
200]
[<matplotlib.lines.Line2D object at 0x9b5952c>]
(-0.059999999999999998, 0.059999999999999998)
Plotting profile collected on 1998-01-22 02:17:00.0

(long lines from original output removed)


[<matplotlib.lines.Line2D object at 0x9c4a50c>]
(1200.0, 0.0)
>>>
>>> xlabel('Oxygen (ml/l)')
<matplotlib.text.Text object at 0x997836c>
>>> ylabel('Depth (m)')
```

```
<matplotlib.text.Text object at 0x9978dac>
>>> title("Mooring M2 Bottle Data: " + dtList[0].split(' ')[0] + " to " + dtList[-1].split(' ')[0])
<matplotlib.text.Text object at 0x9ab57ac>
>>> text(5, 800, 'Blue: Before year 2000\nRed: After Year 2000')
<matplotlib.text.Text object at 0x987ee6c>
>>>
>>> show()
>>>
>>> fig.savefig("M2_oxy.png")
```

Google Earth dynamic queries

This page last changed on May 17, 2010 by mccann.

Here is the setup procedure for creating a dynamic query from a relational database directly into Google Earth (following view-based refresh directions posted at http://code.google.com/apis/kml/documentation/kml_tut.html#network_links).

The following two files are all that is needed to provide the capability viewable at http://dods.mbari.org/uwcd/viewUWCD.kml

1. Create a "master" .kml file with the refresh parameters and a network link to the .cgi file:

```
> pwd
/var/www/dods_html/uwcd
<187 elvis.shore.mbari.org /dods_html/uwcd> cat viewUWCD.kml
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://earth.google.com/kml/2.2
http://code.google.com/apis/kml/schema/kml22beta.xsd">

<Folder>
<name>MBARI Upper Water Column Data</name>
<visibility>0</visibility>
<open>0</open>
<description>Check boxes to see data in your view.
Queries begin 2 seconds after you stop navigating.</description>
<NetworkLink>
<name>BOG Bottle Data</name>
<visibility>0</visibility>
<open>0</open>
<description>Query for Bottle Data</description>
<refreshVisibility>0</refreshVisibility>
<flyToView>0</flyToView>

<Link>
<href>http://elvis.shore.mbari.org/cgi-bin/bogKML.cgi</href>
<refreshInterval>2</refreshInterval>
<viewRefreshMode>onStop</viewRefreshMode>
<viewRefreshTime>1</viewRefreshTime>
<viewBoundScale>0.75</viewBoundScale>
</Link>
</NetworkLink>


</Folder>
</kml>
```

2. Create the .cgi file with the spatially constrained query and KML output:

```
> pwd
/var/www/cgi-bin
<195 elvis.shore.mbari.org /www/cgi-bin> cat bogKML.cgi
#!/usr/bin/perl

# Script to query BOG Bottle data from Google Earth
#
# Mike McCann  16 april 20009
# MBARI

use DBI;

#
# Debugging stuff:  All debugging print statements are commented out with
# '##'.  Must turn on text/html to see the debugging output, should you
# need to do that.
#
```

```
##print "Content-type: text/html\r\n\r\n";
##print "Hello, World.";

##print "String = $ENV{'QUERY_STRING'}\n\n<p>";
@values = split(/&/, $ENV{'QUERY_STRING'});
my %qVars = ();
foreach $i (@values) {
($varname, $mydata) = split(/=/, $i);
$qVars{lc $varname} = $mydata;
##print "The value of $varname is $mydata\n\n<p>";
}

#
# Get parameters from Query String.  GE will provide bbox.  qsql is for
# potential future use.
#

$bbox = $qVars{'bbox'};
$qsql = $qVars{'qsql'};

##print "bbox = $bbox\n";

if ( $bbox ) {
($west, $south, $east, $north) = split(',', $bbox);
}
elsif ( $qsql ) {
# Use this SQL for the query below
}
else {  # Some default bounding box - for testing
($west, $south, $east, $north) = (-124, 35.2, -122, 36);
}

#
# Set up database connection (replace <...> entries with actual connection parameters)
#
##print "\nConnecting to the Database<br>";
my $dbh = DBI->connect("DBI:Sybase:server=<db_server>", "<ro_login>", "<ro_password>",
{PrintError => 1});
$dbh->do("use BOG");

#
# Count the records for the query and set up the stride for subsampling
# - careful that we use the same join and where clause as we do below...
#
my $countSQL = <<EOS;
SELECT     count(*)
FROM       dbo.BCTD INNER JOIN
dbo.EXPEDITION ON dbo.BCTD.EXPD = dbo.EXPEDITION.expd
WHERE     (dbo.EXPEDITION.dec_lat > $south) AND (dbo.EXPEDITION.dec_lat < $north) AND
(dbo.EXPEDITION.dec_long > $west) AND (dbo.EXPEDITION.dec_long < $east)
EOS

##print "\nExecuting countSQL: $countSQL";
my $sth = $dbh->prepare($countSQL);
$sth->execute;
while ( my $rrow = $sth->fetchrow_arrayref() ) {
$numRows = $$rrow[0];
}
##print "<br>numRows = $numRows\n";
$bottlesToDisplay = 500;
$stride = int($numRows / $bottlesToDisplay);
$stride = 1 if $stride < 1;
##print "<br>stride = $stride\n";
```

```
#
# Construct SQL statements with DateTime given in ISO8601 format.  If SQL passed in then
# construct that query with the additional ISO8601 date format.  Apply the stride from above.
#
my $sql;
if ( $qsql ) {          # Add DateTime and put in some newlines for cleaner display in the
description
# Need to add ISO date time to SELECT
$qsql =~ s/FROM/, rtrim(convert(char, CollectionEventDTG,126)) + 'Z' as DateTime \nFROM/;
$qsql =~ s/WHERE/\nWHERE/;
$qsql =~ s/ORDER/\nORDER/;
$sql = $qsql;
}
else {
##SELECT     dbo.EXPEDITION.dec_lat AS Latitude, dbo.EXPEDITION.dec_long AS Longitude,
##           dbo.BCTD.BOTTLE, dbo.EXPEDITION.cruise, dbo.BCTD.DEPTH, dbo.BCTD.CTRB_ID,
$sql = <<EOS;
SELECT     *,
rtrim(convert(char, dbo.EXPEDITION.date_time,126)) + 'Z' as DateTime
FROM       dbo.BCTD INNER JOIN
dbo.EXPEDITION ON dbo.BCTD.EXPD = dbo.EXPEDITION.expd
WHERE     (dbo.EXPEDITION.dec_lat > $south) AND (dbo.EXPEDITION.dec_lat < $north) AND
(dbo.EXPEDITION.dec_long > $west) AND (dbo.EXPEDITION.dec_long < $east) AND
BCTD.CTD_BOTTLE_ID % $stride = 1
ORDER BY  dbo.EXPEDITION.date_time
EOS

}


#
# Execute the SQL, and build the placemarks
#
##print "\nExecuting sql: $sql";
$sth = $dbh->prepare($sql);
$sth->execute;

my $placemarks = '';
my $count = 0;
while ( my $rrow = $sth->fetchrow_hashref() ) {
##foreach my $k ( keys %$rrow ) {
##print "k = $k, v = " . $$rrow{$k} . "\n";
##}
##print "\n<br>";


$lat = $$rrow{'dec_lat'};
$lon = $$rrow{'dec_long'};
$depth = $$rrow{'DEPTH'};

unless ($lat && $lon && $depth) {
next;
}

$isodate = $$rrow{'DateTime'};
$bottle = $$rrow{'BOTTLE'};
$cruise = $$rrow{'cruise'};

$name = "$cruise $bottle";

$placemarks .= "<Placemark>\n<name>$name</name>\n";
$placemarks .= "<TimeStamp>\n<when>$isodate</when>\n</TimeStamp>\n";
$placemarks .= "<description><![CDATA[";
$placemarks .= "<div>$name</div><br><br>\n";
foreach my $k ( sort keys %$rrow ) {
$placemarks .= "<div><b>$k:</b> " . $$rrow{$k} . "</div>\n";
```

```
}
$placemarks .= "]]></description>\n";
$placemarks .= "<styleUrl>#square_pair</styleUrl>\n";
$placemarks .= "<Point>\n<altitudeMode>absolute</altitudeMode>\n";
$placemarks .= "<coordinates>$lon,$lat,-${depth}</coordinates>\n</Point>\n";
$placemarks .= "</Placemark>\n";

$count++;

}
$dbh->disconnect;


#
# Build up KML and deliver it to the client
#
my $kml = <<EOK;
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://earth.google.com/kml/2.1
http://code.google.com/apis/kml/schema/kml21.xsd">


<Folder>
EOK
if ( $qsql ) {
$kml .= "<name>BOG DB Bottle Query</name>\n";
}
else {
$kml .= "<name>Bottles</name>\n";
}

$kml .= <<EOK;
<Document>
<name>Individual Bottles</name>
<Style id="sn_square_copy29">
<IconStyle>
<color>ffffff00</color>
<scale>0.5</scale>
<Icon>
<href>http://maps.google.com/mapfiles/kml/shapes/square.png</href>
</Icon>
</IconStyle>
<LabelStyle>
<scale>0.7</scale>
</LabelStyle>
</Style>
<Style id="sh_square_copy13">
<IconStyle>
<color>ffffff00</color>
<scale>0.66</scale>
<Icon>
<href>http://maps.google.com/mapfiles/kml/shapes/square.png</href>
</Icon>
</IconStyle>
<LabelStyle>
<scale>0.77</scale>
</LabelStyle>
<BalloonStyle>
<bgColor>ffffffff</bgColor>
<textColor>ff000000</textColor>
<text>\$[description]</text>
<displayMode>default</displayMode>
</BalloonStyle>
</Style>
```

```
<StyleMap id="square_pair">
<Pair>
<key>normal</key>
<styleUrl>#sn_square_copy29</styleUrl>
</Pair>
<Pair>
<key>highlight</key>
<styleUrl>#sh_square_copy13</styleUrl>
</Pair>
</StyleMap>

<visibility>0</visibility>
<open>0</open>
<description><![CDATA[$numRows Bottles in this view, showing every
$stride<br><br><br><br>Resulting from query<br><pre>$sql</pre>]]></description>
$placemarks
</Document>
</Folder>
</kml>
EOK

print "Content-type: application/vnd.google-earth.kml+xml\r\n\r\n";
print $kml;
```

installing DRDS and connecting to BOG

This page last changed on May 17, 2010 by mccann.

These steps document the process on setting up the DRDS on a server and then connecting that server (as read-only) to the BOG database.

1. Download dods.war from here: http://opendap.org/pub/dods/DODS-Java-1.1/1.1.7/dods.war
2. Install apache tomcat (tested with 6.0.18)
3. Drop the dods.war in the $TOMCAT_HOME/webapps directory and start tomcat.
4. This will create a dods directory in the webapps directory.
5. Shutdown tomcat
6. Download the JTDS driver from: http://jtds.sourceforge.net/ and install the jar file in $TOMCAT_HOME/lib
7. Edit the $TOMCAT_HOME/webapps/dods/WEB-INF/web.xml file.
    a. Configure the DRDS servlet to connect to the BOG Datbase. The web.xml should look something like:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>


<servlet>
<servlet-name>
dts
</servlet-name>

<servlet-class>
dods.servers.test.dts
</servlet-class>

<init-param>
<param-name>DebugOn</param-name>
<param-value>showRequest showResponse </param-value>
</init-param>

</servlet>

<servlet-mapping>
<servlet-name>dts</servlet-name>
<url-pattern>/dts</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>dts</servlet-name>
<url-pattern>/dts/*</url-pattern>
</servlet-mapping>


<!-- MBARI's BOG Database connection -->
<servlet>
<servlet-name>
bog
</servlet-name>

<servlet-class>
dods.servers.sql.drds
</servlet-class>

<init-param>
<param-name>JDBCdriver</param-name>
<param-value>net.sourceforge.jtds.jdbc.Driver</param-value>
</init-param>
```

```
<init-param>
<param-name>JDBCconnectionURL</param-name>
<param-value>jdbc:jtds:sqlserver://solstice.shore.mbari.org/BOG</param-value>
</init-param>

<init-param>
<param-name>JDBCusername</param-name>
<param-value>read-only-username</param-value>
</init-param>

<init-param>
<param-name>JDBCpassword</param-name>
<param-value>read-only-username-password</param-value>
</init-param>

<init-param>
<param-name>JDBCMaxResponseLength</param-name>
<param-value>100000</param-value>
</init-param>

<init-param>
<param-name>UseDataSetName</param-name>
<param-value></param-value>
</init-param>


<init-param>
<param-name>DebugOn</param-name>
<param-value>showRequest  JDBC</param-value>
</init-param>

</servlet>

<servlet-mapping>
<servlet-name>bog</servlet-name>
<url-pattern>/bog</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>bog</servlet-name>
<url-pattern>/bog/*</url-pattern>
</servlet-mapping>

</web-app>
```

8. Now you need to specify the DDS, DAS, and the HTML template for accessing the data in the BOG database.
   a. Create a directory named 'bog' under the $TOMCAT_HOME/webapps/dods/datasets directory.
   b. Create three directories under the 'bog' directory
      i. das
      ii. dds
      iii. info
9. In the 'das' directory create a DAS file that matches the name of the table you want to connect to. For this example, I connected to the BCTD table so I created a file called 'BCTD'.

```
Attributes {
BCTD {
CTD_BOTTLE_ID {
}
CRUISE {
}
EXPD {
}
SEQ {
}
DEPTH {
}
```

```
BOTTLE {
}
PROJECT {
}
CTRB_ID {
}
PLATFORM {
}
RJDAY {
}
DATE_TIME {
}
JTIME {
}
YEAR_DATE {
}
DEC_LAT {
}
DEC_LONG {
}
SITE_ID {
}
PRESSURE {
}
CHL_GFF {
}
PHAEO_GFF {
}
FOFA_GFF {
}
VF_GFF {
}
VE_GFF {
}
SENS_GFF {
}
FO_GFF {
}
FA_GFF {
}
FLUOR_RTO {
}
CHL_1u {
}
PHAEO_1u {
}
FOFA_1u {
}
VF_1u {
}
VE_1u {
}
FO_1u {
}
FA_1u {
}
SENS_1u {
}
CHL_5u {
}
PHAEO_5u {
}
FOFA_5u {
}
VF_5u {
}
```

```
VE_5u {
}
SENS_5u {
}
FO_5u {
}
FA_5u {
}
TMP {
}
SAL {
}
CONDUCT {
}
SIG_T {
}
PARCOS {
}
PAR4PI {
}
TRANSMISS {
}
CHLA {
}
FLUOR {
}
NH4 {
}
NO2 {
}
NO3 {
}
SIO4 {
}
PO4 {
}
TCO2 {
}
O2 {
}
OXY_ML {
}
OXY_PS {
}
IRRD_490 {
}
IRRD_520 {
}
IRRD_555 {
}
ALTIMETER {
}
rdep {
}
sal2 {
}
temp2 {
}
cond2 {
}
pot_tmp {
}
OxyT {
}
OxyC {
}
```

```
TransV {
}
TransBeam {
}
POT_TMP2 {
}
OxyV {
}
ISUS_NO3 {
}
}
}
```

10. In the dds directory, create a DDS file that is also named after the table you will connect to. For this example, the file name is BCTD again and looks like (unlike the DAS file, this file should have no extension, i.e. it should be named just 'BCTD'):

```
Dataset {
Sequence {
Int32 CTD_BOTTLE_ID;
String CRUISE;
Int32 EXPD;
Int32 SEQ;
Int32 DEPTH;
Int32 BOTTLE;
String PROJECT;
String CTRB_ID;
String PLATFORM;
Int32 RJDAY;
String DATE_TIME;
Float64 JTIME;
Int32 YEAR_DATE;
Float64 DEC_LAT;
Float64 DEC_LONG;
String SITE_ID;
Float64 PRESSURE;
Float64 CHL_GFF;
Float64 PHAEO_GFF;
Float64 FOFA_GFF;
Float64 VF_GFF;
Float64 VE_GFF;
Float64 SENS_GFF;
Float64 FO_GFF;
Float64 FA_GFF;
Float64 FLUOR_RTO;
Float64 CHL_1u;
Float64 PHAEO_1u;
Float64 FOFA_1u;
Float64 VF_1u;
Float64 VE_1u;
Float64 FO_1u;
Float64 FA_1u;
Float64 SENS_1u;
Float64 CHL_5u;
Float64 PHAEO_5u;
Float64 FOFA_5u;
Float64 VF_5u;
Float64 VE_5u;
Float64 SENS_5u;
Float64 FO_5u;
Float64 FA_5u;
Float64 TMP;
Float64 SAL;
Float64 CONDUCT;
Float64 SIG_T;
Float64 PARCOS;
Float64 PAR4PI;
```

```
Float64 TRANSMISS;
Float64 CHLA;
Float64 FLUOR;
Float64 NH4;
Float64 NO2;
Float64 NO3;
Float64 SIO4;
Float64 PO4;
Float64 TCO2;
Float64 O2;
Float64 OXY_ML;
Float64 OXY_PS;
Float64 IRRD_490;
Float64 IRRD_520;
Float64 IRRD_555;
Float64 ALTIMETER;
Float64 rdep;
Float32 sal2;
Float32 temp2;
Float32 cond2;
Float64 pot_tmp;
Float64 OxyT;
Float64 OxyC;
Float64 TransV;
Float64 TransBeam;
Float64 POT_TMP2;
Float64 OxyV;
Float64 ISUS_NO3;
} BCTD;
} test;
```

11. If you create a file named BCTD.html and put it in the info directory, it will provide some HTML for the user that will give them more information about the BOG dataset. For example:

```
<h2>BOG Dataset</h2>
<h3>
<p>
This is the MBARI BOG dataset
</p>
<p>
Please go to the following for more information:
</p>
<p>
<a href="http://www.mbari.org/bog/"     > MBARI's Biological Ocean Group </a>
</p>
</h3>
```

12. In the info directory create a file named dods.servers.sql.drds.html with the HTML show below. This HTML will show up in an information page that users will see when they browse to the DRDS service for your BOG dataset.

```
<h2>DODS Relational Database Server</h2>

<p>
This data is being served by a DODS Relational Database Server (DRDS), which
is a somewhat unusual instance of a DODS server. All of the data served by the
DRDS is stored in a relational database management system (DBMS). Queries from
DODS clients are translated into SQL queries and sent to the underlying DBMS.
The reuturned data is read by the DRDS and returned to the requesting client.
</p>

<p>
Because the DRDS is essentially a front end to a DBMS, the DDS's have a somewhat
different meaning than in other DODS servers. Each DDS served by the a DRDS
represents a table in the underlying DBMS. Since queries to the DBMS will return
an unknown amount of data, each DDS basically must contain a representation
of the tables contents defined as a DODS <b>sequence</b>. Because of this it
rarely makes sense for a client to request the entire &amp;quot;dataset&amp;quot;, as
```

```
this request will return the entire contents of the table. When requesting data
from a DRDS, it is best to get the dataset information (using the .info extension
on the DODS URL) and then build a constrained request that just returns data
that is actually desired. This doesn't mean that the entire table cannot be
requested and sent, it is simply a caution that each dataset/DDS/table may in
fact be very large.
</p>

<h3>Server Functions:</h3>
<p>
<b>unique():</b> This function gets used in the selection part of the DODS
constraint expression. It requires no parameters. Calling this funcion will
cause the return to contain only unique rows. This is useful for sifting
through numerous identical fields, for example station data that contains
instrument names. This is NOT very useful if you are trying to retrieve a
ship track...
<br>Example:
<pre>
http://nasty.dods.url/server/dataset.dods?var1,var2&unique()
</pre>
</p>


<p>
<b>Regular Expressions:</b> The typical DODS server supports the full range
regular expression syntax. THIS DODS server does not. Limited string matching is
available. The wild card characters '.' and '.*' may used. The '[...]' notation
for matching a range of characters may be used, but only to match a single
character to a range of possiblities.
<br>Examples:
<pre>
http://nasty.dods.url/server/dataset.dods?var1,var2&var3~=".*south.*"
http://nasty.dods.url/server/dataset.dods?var1,var2&var3~=".south."
</pre>
</p>

<p>
<h3>NOTE:</h3> String fields in the DBMS may contain white space (spaces). In order
for a constraint to match you must include the spaces. For example, if you
wish to retrieve all data where a field called <i>location</i> has a value
of "Southern Ocean", in your DODS URL you will have to represent the space
between "Southern" and "Ocean" with the symbol "<b>%20</b>". Like this:
<pre>    location="Southern%20Ocean"</pre>
However, if the DODS URL is formed with a space like this:
<pre>    location="Southern Ocean"</pre>
It will not be handled correctly by the DRDS.
</p>

<p>Have fun!</p>
```

13. Now start up Tomcat and browse to http://localhost:8080/dods/bog and you will see your table with some links that you can explore.

Pros

1. Nice way to make a RDB look like a DODS server.

Cons

1. Not active (most recent release is 2004)
2. Will not compile against Java 1.5(note says that although I did not verify)
3. Not sure if I can make the backing RDB into a Grid dataset instead of just a Sequence.

Installed on elvis using above instructions on 18 May 2009

Example query for bottles with depths > 2000:

http://dods.shore.mbari.org:8080/dods/bog/BCTD.ascii?&BCTD.DEPTH%3E2000

Project kick-off meeting

This page last changed on Jan 15, 2009 by mccann.

Statements of what success looks like for this project:

1. Having the data sets organized with links on one page. Having access to the data would be great.
2. A way to access database data directly from multiple platforms avoiding 'out-of-sync' issues.
3. Creation of a Matlab toolbox for loading, analyzing, and visualizing Climate Forecast formatted data accessible from OPeNDAP servers.
4. Consistent interface to data so that applications may be easily built.
5. Geared toward science use. Need to match access interface to tools that people use.
6. Stated in the form of a Use Case: Able to construct a climatology from ROV, AUV, BOG, and Mooring CTD data without too much fuss.
7. Be able to go to a single server and find a tree of data where each leaf or branch has a standard way of accessing the data.
    - This is programmatic access where libraries are available to developer to build applications to access all the data
    - Preference for Unidata's Common Data Model and THREDDS servers
8. It's be nice to have data browse capability along with programmatic access.
9. Being on a warm beach with a cold beer in hand.

Notes for 14 Jan 2009 meeting

Attending: Mike McCann, Fred Bahr, Kevin Gomes, Brian Schlining, Reiko Michisaki, Mike Godin, Francisco Chavez, John Ryan

- Review of above statements

John will get back to me with his use cases. Submitted by email

- Review of Work Breakdown Structure

Moved Brian's 40 hours to the R2 release target making the "Develop Matlab toolkit for using THREDDS accessible data" item task 120 hours for delivery by R2: 31 July 2009.

Need for a Systems Engineering Diagram identifying data sets, their hosts, servers, services and those reponsible stated. Francisco and Reiko will work on this and post it in this space.

- Meeting schedule (weekly, monthly, never?)

Monthly, about in the middle of the month. Mike will make announcements

John Ryan's 2 Use Cases

This page last changed on Jan 15, 2009 by mccann.

-----Original Message-----
From: John Ryan [mailto:ryjo@mbari.org]
Sent: Wednesday, January 14, 2009 10:53 AM
To: McCann, Mike
Cc: Chavez, Francisco
Subject: RE: One-Stop Shopping for MBARI Upper Water Column Data kick-off

Hi Mike,

You are familiar with some of my use cases, e.g. being able to extend mooring data queries/plotting across historical QC'd deployments and the current deployment. More recently, I have been thinking about use cases for ship and AUV data:
1) AUV
The AUV quicklook plots and data serve well for knowing when, where and what, and permitting analysis by various software, and I think a relatively simple front-end would facilitate access greatly. For example, I am picturing a calendar top end, in which the user can page through year and month and see AUV survey days highlighted. The links from those days bring you to the 2-column quick-look, because this plot does the best job of showing where and what for that survey. Then below that plot are the links to other quicklook plots (QC, nav) and data files in different formats.
2) Ship
Similarly, I like the idea of calendar/quick-look plots as a top-end organizer. For example, say I wanted to look at what was happening at M1 during June 2008. I could tune my calendar to June 2008, see a monthly time series from the M1 sensors, select June X on which the Lobos conducted a profile and drill down into full-resolution CTD profiles, and physical/chemical/phytoplankton results from bottle samples. For the CTD profiles, I like one thing the CIMT data site does, which is to show in a quick-look plot the current profile overlaid on a background of all profiles taken at that site, so the user can readily see anomalies. There are a number of ways to do this, e.g. showing a profile and its anomaly profile (relative to the climatology for that month or season), etc.

Some initial thoughts...

John

Project meeting on 10 September 2009

This page last changed on Sep 10, 2009 by mccann.

Notes from Project meeting 10 September 2009 1000-1130 Bldg G Conference Room

Attending: Mike McCann, Francisco Chavez, John Ryan, [~reiko], Luke Beatman, Fred Bahr

2010 Data Visualization proposal feedback

In essence the feedback is:

For Phase 2 please address the following:

- *Provide a status on the current state of One Stop Shopping project as well as projections of what will or will not be accomplished in 2009.*
- *Clarify what will specifically be developed, requiring the 1 FTE total for 2010.*
- *Clarify what additional capabilities this work will provide to the institute.*
- *Clarify why these addition capabilities are needed, who will be able to use them, and what data sets will be accessible for display with the new plotting/visualization capabilities*.

We spent about 30 minutes discussing our response and came up with these items which will be included an additional 2+ pages to the written proposal:

1. Focus on the CANON project (901009) which references using the infrastructure produced by the One-Stop Shopping and AOSN projects:
"*To capture and integrate observed data in real-time, we will utilize infrastructure developments made as part of the "One-stop shopping proposal (901015) and collaborative tools developed under the AOSN proposal (900738).*"
2. Emphasize that what we propose is linked with other efforts including CenCOOS, CANON, OceanSITES and that for the investment of 1 FTE software engineer we will also reap benefits from work provided by staff in these other groups.
3. State clearly that the One-Stop Shopping project finishes with providing greater efficiency for the programmer, but not necessarily for the end user of the data

In specific responses to our average and below average scores:

Criteria	Our response
Uniqueness: How unique is this contribution?	MBARI has a unique collection of platforms and instruments that are routinely deployed to measure upper ocean parameters. Providing access and visualization capabilities to the data sets produced from these deployments is transitively unique.
Timeliness: Why should this project move forward now? What are the drivers?	The One-Stop Shopping project finishes with providing greater efficiency for the programmer, but not necessarily the end user of the data. The 2010 Visualization project finishes what the 2009 One-Stop Shopping project started by implementing user friendly front-end to our data archives. With MBARI's implementation and operation of SSDS - producing richly described data sets - we have the opportunity to build upon that success and demonstrate and end-to-end ocean observatory data system with next year's CANON project.
Prior Productivity: Has the project leadership been successful with prior support on this project?	The project team members and leadership has been successful with previous MBARI data access portals, namely the Expedition Database and the Samples Database development and maintenance efforts. The team has also been responsible for

	the development and implementation of the Shore Side Data System for MOOS, OASIS, and AUVCTD data streams.
Does the cost benefit analysis show that this project is well justified?	We can perform an analysis of the various technologies and packages and implement the one (or two) that have the best cost/benefit ratio. In terms of evaluating the cost/benefit of the overall project we need to estimate the MBARI's expenditure in it's development and operation of all it upper water column measurement programs for an estimate of the cost term. In the case of projects such as CANON there is no data management component within the project - that project is depending on existing MBARI data management infrastructure (and this project) to make the data available in an integrated way. An additional cost is the expenditure for the proposed work in 2010 (1 FTE engineer and some funds for training). The benefit will be more efficient and interoperable access to data that are important to the success of several projects that depend on upper water column data. This is something that is difficult to assign a dollar value but maybe something easier to assign an subjective 'lack of frustration' value, or a quickness of data report preparation value.
Will the effort benefit a large number of users?	Yes. All of the research groups at MBARI that do work in the upper ocean will benefit as will CenCOOS and the other exteranal groups that we collaborate with such as OceanSITES, and the OOI. External users such as CenCOOS are already using the products developed by the One-Stop Shopping project and. We expect the relationship to continue next year as we add capabilities to the foundation developed in 2009.
How compelling are other justifications for this project?	There is really no other institution that combines an operational capacity and a mature observational data management system such as SSDS. We intend to leverage this capability to provide a unique data access and visualization system built upon community standards and conventions.

Mike will work on revision to the to the proposal text an distribute a draft by tomorrow (Friday) evening. Edits to the draft need to be sent back to Mike by Monday morning 14 September 2009.

Data Access web page mock-up

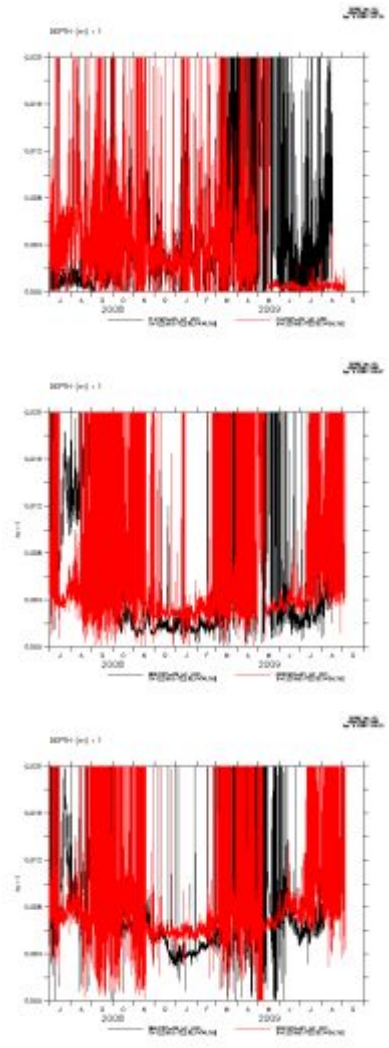
Reiko showed her overall data access page which is a presented as a matrix with axes of platforms and parameters. The links in the cells should take a user to straight-forward pages that show how or provide access to the data (in many cells we have a lot of work to do).

THREDDS server status

SSDS managed HS2 data from M1 and M2 (2004 to the present) were added to the current catalog. This was done in response to Sergey's question whether the correct calibrations have been applied to all the deployments. We discovered that in fact the most recent deployments were not using the most recent calibrations and that oversight was fixed. Inter-comparison of the multiple deployments is now simplified by using TDS, e.g. here is a ferret script that plots the 3 HS2 variables from M1 and M2 over several deployments and still showing some issues with the data:

```
use "http://elvis.shore.mbari.org/thredds/dodsC/moorings/OS_M1_HS2"
```

```
use "http://elvis.shore.mbari.org/thredds/dodsC/moorings/OS_M2_HS2"
set reg/t=1-jul-2008:9-sep-2009
plot/vlimits=0:0.02 FL676[d=1],FL676[d=2]
plot/vlimits=0:0.02 BB676[d=1],BB676[d=2]
plot/vlimits=0:0.02 BB470[d=1],BB470[d=2]
```



Black lines are from M1, Red lines are from M2

The NetCheck program from Coastwatch program has been installed on pismo to periodically confirm that the data are accessible through the tomcat server on elvis. Email is currently being sent to Mike if there are any problems.

Web access portal for BOG data

Francisco asked for Reiko and Luke to think about this task. Mike discussed how MBARI could develop a web site similar to the Samples Database or Expedition Database query pages that not only provides access to the BOG relational database data, but also to the netCDF data (mooring, AUVCTD, Glider) that is available via OPeNDAP and TDS. Mike's preferred programming language is Python. It is a rapid development scripting language with powerful object orientated capabilities and a rich set of libraries available for relational database access and OPeNDAP data access (see the PyDAP example that access BOG bottle data from deeper than 4000 m) as well as output to various formats such as NetCDF, KML, simple text and custom graphics (see PyNGL PyNIO and matplotlib). A simple page providing access to BOG data can be developed in 2009 with an eye to extending the query to the netCDF data in 2010 should the Data Visualization project be approved.

RAMADDA

Unidata's RAMADDA server (just now in alpha release) was not examined. Instead we spent some time looking at how it would be to access and visualize upper water column data using Google Earth - as documented in the Setup for Google Earth dynamic queries miscellaneous document in this space.

Project meeting on 12 November 2009

This page last changed on Nov 13, 2009 by brian.

Notes from Project meeting 12 November 2009 1330-1230 Surface Commons Conference Room

Attending: Mike McCann, Francisco Chavez, [~reiko], [~rich], Brian Schlining, Fred Bahr

Since our last meeting this response to Phase 2 feedbackwas submitted to the management team for our 2010 proposal.

Review of approved labor and expense budgets: 1280 Engineering hours (Mike, Rich, Kevin) and budget for travel, training, and software.

Agenda

- Data Access web page mock-up (Reiko)
  Reiko will add link to thredds server on elvis (with caveat that it is for programmatic access) and distribute the link to the page to the rest of the BOG group.
- THREDDS server status (Mike)
  Still need to stitch together archived QC data with current deployment data.
- Port of ncarchive.jsp to use netcdf-java 4 (Mike)
  Debugging issues with TDS access right now. The plan is to add another access link next to the OPeNDAP access link on the TDS.
- Matlab nctoolbox (Brian)
  Did tech transfer to UCSD/SDCS to make the nctoolbox part of the OOC-CI.
  For COARDS-compliant data you can now subset along coordinate axes.
  Should also add to OPeNDAP site: http://opendap.org/faq/whatClients.html.

  > Brian Schlining sent a request on 2009-11-12 to the OPeNDAP folks to add nctoolbox to their site.

  John Ryan reports that he's used successfully nctoolbox to access satellite, model, and mooring data and it is now his standard tool for accessing netCDF/OPeNDAP data.
- Web access portal for BOG data
  Mike will be working on this the remaining of 2009. Google Earth output will be one of the products.
- 2010 Data Visualization project goals
  We need to coordinate with SAIC, CeNCOOS, and the other regional associations and other groups serving and creating products of upper water column data. Example sites include:[http://whpo.ucsd.edu/][http://www.nanoos.org/nvs/nvs.php?path=NVS-Assets]
  As the specific goals of the 2010 project are not explicitly stated in the proposal we seek input from data users on what interfaces are most useful.

Project meeting on 19 February 2009

This page last changed on Feb 20, 2009 by mccann.

Notes for 18 Feb 2009 meeting 0930-1030 Bldg G Conference Room

Attending: Fred Bahr, Francisco Chavez, Kevin Gomes, Mike McCann, Reiko Michisaki

Systems engineering diagram review

- Need to add ROVCTD
- Would like to have no dotted lines for the BOG data flow

Progress reports

- Mooring
  Fred and Mike reviewed 4 possible ways to combine QC'ed mooring data with currently deployed data:

1. Conversion program to transform existing QC'ed netCDF files to OceanSITES format that SSDS creates
2. Use NCML and THREDDS to alter metadata of existing QC'ed netCDF files
3. Reprocess QC'ed data from Matlab .mat archive
4. Publish all the original observed data to SSDS and use the DPforSSDS software to generate netCDF files

(We prefer option #1 and will proceed.)

- AUVCTD
  Entire archive of decimated Data is available now via THREDDS. How applications can make use of the data is to be determined.
- ROVCTD
  Kevin is investigating DRDS
- BOG
  Kevin is investigating DRDS and will see Reiko about access to BOG after understanding how DRDS works.

Application Discussion

- Initially use Google Earth 5.0 as a geospatial-temporal data browser for data served through the THREDDS server.

Project meeting on 2 April 2009

This page last changed on Aug 24, 2009 by mccann.

Notes for 2 April 2009 meeting 1000-1100 Bldg G Conference Room

Attending: Fred Bahr, Kevin Gomes, Mike McCann

Review of THREDDS server data offerings: http://elvis.shore.mbari.org/thredds/catalog.html

- Mooring TS data are aggregated and available through the server
- For the examples below use the 9 April 2009 catalog: http://elvis.shore.mbari.org/thredds/catalog_090409.html
- Entire archive of decimated Data is available now via THREDDS.
- How applications can make use of the data is to be determined: Can Matlab & loaddap load and use the data?
  Not really:
  - Ferret cannot load all the aggregated irregular time data for a mooring:

```
<103 elvis.shore.mbari.org /u/mccann> ferret
NOAA/PMEL TMAP
FERRET v6.13 (beta)
Linux(g77) 2.4.21-32 - 08/28/08
2-Apr-09 09:52

cancel mode journal
sp rm -f ferret.jnl

yes? use "http://elvis.shore.mbari.org/thredds/dodsC/moorings/OS_M1_TS_090409"
**TMAP ERR: limit on storage for coordinates has been reached
MAX=      750000
```

  - Matlab's loaddap also fails with an I/O error when trying to load all the mooring data:

```
>> addpath('/usr/local/bin');     % For the loaddap command on elvis
>> loaddap('http://elvis.shore.mbari.org/thredds/dodsC/moorings/OS_M1_TS_090409')

Reading: http://elvis.shore.mbari.org/thredds/dodsC/moorings/OS_M1_TS_090409
Constraint:
Error: Network I/O error. Could not read packed array data.
This may be due to a bug in libdap or a problem with
the network connection.
```

  - However; a single variable (TEMP) can be loaded:

```
>> loaddap('http://elvis.shore.mbari.org/thredds/dodsC/moorings/OS_M1_TS_090409?TEMP')

Reading: http://elvis.shore.mbari.org/thredds/dodsC/moorings/OS_M1_TS_090409
Constraint: TEMP
Server version: apache-coyote/1.1
Creating matrix TEMP (528789 x 11 x 1 x 1) with 5816679 elements.
Creating vector TIME with 528789 elements.
Creating vector DEPTH with 11 elements.
Creating vector LATITUDE with 1 elements.
Creating vector LONGITUDE with 1 elements.

and plotted against epoch seconds:
>> T=squeeze(TEMP.TEMP);   % Remove singleton lat & lon dimensions
>> bindx=find(T < 0);      % Find missing values
>> T(bindx)=NaN;           % Set them to NaN
>> offset=datenum(1970,1,1,0,0,0);                 % Epoch second offset in Matlab's
datenum
>> plot(TEMP.TIME/24/60/60 + offset, T(1,:))       % Plot Surface temp for the whole
time series
>> datetick('x', 2)                                % Give a reasonable time axis
```

  - loaddap fails to load the data with a stride specification (this looks like a bug):

```
>> loaddap('http://elvis.shore.mbari.org/thredds/dodsC/auv/dorado_ctd?
latitude[1:2:4000000]')
```

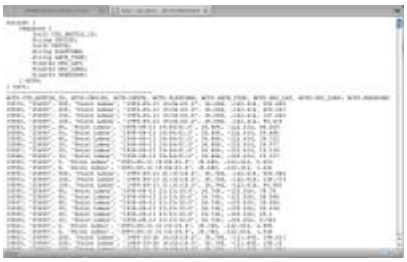
```
Reading: [http://elvis.shore.mbari.org/thredds/dodsC/auv/dorado_ctd]
Constraint: latitude[1:2:4000000]
Error: An internal error was encountered in Vector.cc at line 587:
The server sent declarations and data with mismatched sizes.
Please report this to support@opendap.org
```

notloaddap

Brian Schlining checked a project into subversion that uses NetCDF-Java 4.0 as the data access library in matlab. Here's some examples based on the above data:

```
>> ds = ncdataset('http://elvis.shore.mbari.org/thredds/dodsC/moorings/
OS_M1_TS_090409')
>> temp = ds.data('TEMP');
>> t  = ds.data('TIME');
>> plot(t, temp)
>> datetick('x', 2)
```

- Possible ways to address the above problems include
  1. Creating separate catalogs of the aggregated 10-minute data and the telemetered hourly data
  2. Creating an hourly gridded product from the 10-minute data and aggregating that with the hourly telemetered data, this would still be an irregular grid, but with the fewer points could be read by Ferret
  3. Submitting a bug report regarding the loaddap stride problem.
- BOG
  - Kevin has configured DRDS on his laptop and has demonstrated ability to query the data directly from the database:



- ROVCTD data
  - All of the ROVCTD data is in the Expd Relational database and can be served with the same technique as we'll use for BOG.

Client tools for data access

- Ferret
- Matlab with loaddap
- Google Earth 5.0 as a geospatial-temporal data browser for data served through the THREDDS and other servers.

Project meeting on 7 May 2009

This page last changed on May 07, 2009 by mccann.

Notes for 7 May 2009 meeting 1400-1500 Ship's View Conference Room

Attending: Fred Bahr, Kevin Gomes, Mike McCann, Brian Schlining

Progress on THREDDS server (Mike)

The THREDDS server now has a separate catalog for the examples presented at the last meeting and a current catalog that we're working on now:

The current (working) catalog separates the post-recovery from the SSDS managed telemetered data:

Here is an example (using Ferret) of using a link from the latter to list all the TS data for M1 from 2004 until today:

```
yes? use "http://elvis.shore.mbari.org/thredds/dodsC/moorings/OS_M1_TS"
yes? list TEMP
VARIABLE : Hourly sea_water_temperature (celsius)
DATA SET : MBARI Mooring M1 CTD String Aggregation
FILENAME : OS_M1_TS
FILEPATH : http://elvis.shore.mbari.org/thredds/dodsC/moorings/
SUBSET   : 11 by 202 points (DEPTH (m)-TIME)
LONGITUDE: 122W(-122)
LATITUDE : 36.8N
1      10     20     40     60     80    100    150    200    250    300
1      2      3      4      5      6      7      8      9     10     11
21-OCT-2004 14:30 /   1:   9.60
9.78   ....   ....   ....   ....   ....   ....   ....   ....   ....
21-OCT-2004 15:30 /   2:  11.86
10.94   ....   ....   ....   ....   ....   ....   ....   ....   ....
21-OCT-2004 16:30 /   3:  15.09
12.30   ....   ....   ....   ....   ....   ....   ....   ....   ....
21-OCT-2004 17:30 /   4:  15.05
13.31   ....   ....   ....   ....   ....   ....   ....   ....   ....
21-OCT-2004 18:30 /   5:  14.53  14.24   ....  14.13  14.03  13.09  12.73   9.91   9.34   8.55
8.62
21-OCT-2004 19:30 /   6:  14.64  14.27   ....  14.11  14.02  13.13  12.63  10.14   9.39   8.61
8.20
21-OCT-2004 20:30 /   7:  14.88  14.26   ....  14.04  13.89  12.92  12.48  10.06   9.25   8.55
8.08
....

07-MAY-2009 13:30 / 38989:  11.31  11.29   ....   ....   9.11   9.00   8.90   ....   8.13   7.88
7.39
07-MAY-2009 14:30 / 38990:  11.43  11.30   ....   9.51   9.10   9.00   8.86   8.48   8.09
7.83   ....
07-MAY-2009 15:30 / 38991:  11.42
10.45   ....   ....   ....   ....   ....   ....   ....   ....   ....
```

```
07-MAY-2009 16:30 / 38992:  11.43   9.87   ....   9.20   8.97   8.83   8.67   8.21   8.07   ....
7.22
07-MAY-2009 17:30 / 38993:  11.51
10.45   ....   ....   ....   ....   ....   ....   ....   ....   ....
07-MAY-2009 18:30 / 38994:  11.59
10.15   ....   ....   ....   ....   ....   ....   ....   ....   ....
```

Core Data - ncarchive.jsp (Mike)

Mike will work on adapting this web application for working against the THREDDS catalog for delivering text files from the OPeNDAP URLs. [Note: will need to compare with erddap capabilities...]

Progress on BOG data access (Kevin)

Kevin will clean up the documentation and Mike will install on elvis. We will see how useful the sequence data are from clients such as Matlab. Kevin will investigate configuring DRDS to deliver the data as grids instead of sequences.

Matlab OPeNDAP-CF toolkit (Brian)

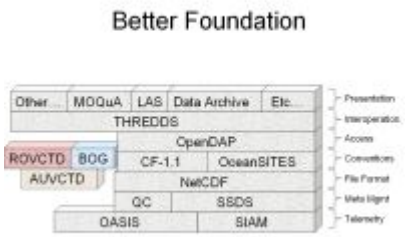
Brian will start working on it and will work with Reiko and John Ryan on delivering a toolkit that meets their needs.

Systems Engineering Information

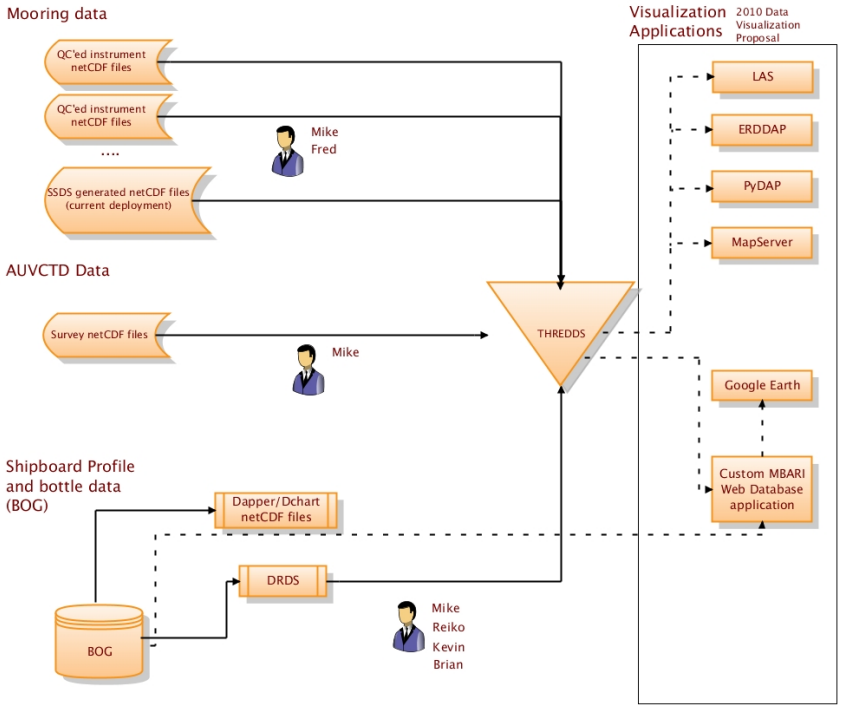
This page last changed on Feb 06, 2009 by mccann.

Implementation Plan

The project proposal outlines the architecture for a better foundation for access to MBARI Upper Water Column Data as indicated in the figure below. A Work Breakdown Structure in the proposal identifies specific release milestones and hours assigned to each project team member. To elaborate further on these milestones an overall diagram is provided below which shows the existing data stores, the path to THREDDS server access and who is responsible for implementing such access.



THREDDS implementation



Upper Water Column Data

This page last changed on Feb 03, 2009 by reiko.

MBARI DATA ACCESS

DATASETS

Platform	Access	*Comments*
Shipboard CTD	1. MS SQL	*On Solstice, BOG database*
Mooring	1. SSDS: a. SIAM Raw data access page b. GetOriginalDataServlet c. Explorer d. QC M0 (plot,data,netcdf) e. QC M1 (plot,data,netcdf) f. QC M2 (plot,data,netcdf) 2. OASIS data archive (web) 3. Live Access Server 4. Real time, netcdf from DODS/SSDS 5. Loboviz 6. CIMT web pages (http) 7. CIMT dods access	# 1. a. http://new-ssds.mbari.org:8080/ssds/siamRawDataStep1.jsp b. http://new-ssds.mbari.org:8080/servlet/GetOriginalDataServlet c. http://new-ssds.mbari.org:8080/ssds/ d. http://dods.mbari.org/data/ssdsdata/deployments/m0/current_qcPlots.html e. http://dods.mbari.org/data/ssdsdata/deployments/m1/current_qcPlots.html f. http://dods.mbari.org/data/ssdsdata/deployments/m2/current_qcPlots.html 2. http://dods.mbari.org:8085/mbariMoos/ncarchive.jsp 3. http://dods.mbari.org/lasOASIS/ 4. *Real time current deployment:* http://dods.mbari.org/cgi-bin/nph-nc/data/ssdsdata/deployments/

		m0/200706/m0_ctd0001_20070621_original.nc.html 5. http://dods.mbari.org/cgi-bin/nph-nc/data/ssdsdata/deployments/ 6. http://www.mbari.org/lobo/loboviz.htm 7. http://new-ssds.mbari.org:8080/cimt/cimt.jsphttp://dods.mbari.org/cgi-bin/nph-nc/data/ssdsdata/deployments/#
AUVCTD	1. SSDS: Files on Tornado 2. Web 3. Mission Logs (http) 4. Mission Logs (dods) 5. Netcdf files (http) 6. Netcdf files (dods) 7. Ryan Lab processed files (http, netcdf, mat, images, odv, logs) 8. Ryan Lab processed files (dods)	1. *Tornado/auvctd Netcdf, mat, odv files* 2. http://dods.mbari.org/data/auvctd/surveys/ 3. http://new-ssds.mbari.org/auvctd/missionlogs 4. http://dods.mbari.org/cgi-bin/nph-nc/data/auvctd/missionlogs/ 5. http://new-ssds.mbari.org/ssdsdata/ssds/generated/netcdf/files/ssds.shore.mbari.org/auvctd/missionlogs/ 6. http://dods.mbari.org/cgi-bin/nph-nc/data/ssdsdata/ssds/generated/netcdf/files/ssds.shore.mbari.org/auvctd/missionlogs/ 7. http://new-ssds.mbari.org/auvctd/surveys/2008/netcdf/http://dods.mbari.org/cgi-bin/nph-nc/data/auvctd/surveys/#
ROV-CTD	1. Samples DB, web access	1. http://mww.mbari.org/

	2. Web ARCHIVES	samplesDB/queries/ 2. http://search.mbari.org/ARCHIVE/rovctd/
Glider 014	1. Web access via SPRAY data page	*Line 67* MontereyBay (Scripps: http://spray.ucsd.edu/cgi-bin/archive_init.pl?start_archive=ARCHIVE+DATA)
NOEP	1. NOEP Web 2. MS SQL	*On FOG, BLS, Market,*

PLATFORMS

Platform	Resolution	Location	Depths	Profiles
Ship (Time series)	Monthly	C1, M0, M1, M2	Surface and profiles	Discrete bottle depths, profiles in meter bins
Ship (SECRET)	Quarterly	CalCOFI Stations	Surface and profiles	Discrete bottle depths, profiles in meter bins
AUV	Monthly	Various MontereyBay	Surface and profiles	Profiles in meters; yoyo surveys
ROV	Mission	Various MontereyBay	Surface and profiles	Various depths
Glider	Mission	CalCOFI Line 67	Surface and profiles	Yoyo surveys
Mooring	10 minute, hourly, daily	M0, M1, M2	Surface and profiles	Discrete depths 0, 10, 20, 40, 60, 80, 100, 150, 200, 250, 300m

SAMPLES

Parameter	Platform				
	Ship	**Mooring**	**AUVCTD**	**ROV**	**Glider**
Date Range	Mar 1989-present	Mar 1989-present	Apr 2003 - present	Oct 1988 - present	Apr 2007 - present
Frequency	Time series: Monthly	10 min, hourly, daily	Missions occur monthly	Missions at irregular intervals	Missions quarterly

	SECRET: Quarterly (Line 67)				duration 90 days
Type	CTD Bottle, Profiling	Surface CTD at fixed depths	Yo-yo surveys	Survey follows the ROV	Yo-yo surveys
Depth	Bottle 0,5,10,15,30,40,60,80,100,150,200m Profiling 0-200 (1 m increments)	M0: 0, 10, 20, 40, 55m M1: 0,10,20,40,60,80,100,150,200,250,300m M2: 0,10,20,40,60,80,100,150,200,250,300m	0-200m	Irregular depths	0-200 m (5 m increments)
Location(map)	C1, Mooring1, Mooring2, CalCOFI Line 67	M0 M1 M2	CalCOFI Line 67	Various locations in MontereyBay	CalCOFI Line 67
Temperature	Ship	Mooring	AUVCTD	ROV	Glider
Conductivity	Ship	Mooring			
Salinity	Ship	Mooring	AUVCTD	ROV	Glider
Transmissivity	Ship			ROV	
Oxygen	Ship	Mooring? Optode? surface	AUVCTD	ROV	
Fluorescence	Ship	Mooring, surface	AUVCTD		Glider
Density	Ship				
Nitrate	Ship		AUVCTD		
Bioluminescence			AUVCTD		
BBP at 676 & 470			AUVCTD		
Metsys		Mooring, surface			
Air Temperature		Mooring, surface			
Winds (speed, direction)		Mooring, surface			
PCO2					
ADCP		Mooring, surface			
PRR		Mooring, surface			

This page last changed on Feb 11, 2009 by mccann.

Upper Water Column Data access web page

Mooring

Quality Controlled Data (1989 to 2007)	Current Deployment Data (2007 to present)
Live Access Server (data subsetting and graphics) Core Mooring Access (textual data extraction)	Shore Side Data System (netCDF files and plots)

Aggregated Data from SSDS (2004 to present)
THREDDS (Sea Water Temperature and Salinity)

AUVCTD

Aggregated Data (2003 to present)
THREDDS (All decimated netCDF survey data)

ROVCTD

BOG

.bookmarks

This page last changed on Nov 19, 2010 by .

One-Stop Shopping for MBARI Upper Water Column Data Bookmarks

This page is a container for all the bookmarks in this space. Do not delete or move it or you will lose all your bookmarks.
Bookmarks in One-Stop Shopping for MBARI Upper Water Column Data | Links for One-Stop Shopping for MBARI Upper Water Column Data


Bookmarks

There are no bookmarks to display.